Title: Action Classification

Project Team Members:

1. Sri Rupin Potula
2. Vibha Swaminathan
3. Vibhu Dixit
4. Abhishek Mulasi

Abstract:

The main objective of phase -1 is to extract the features from images. In order to do this, the hmdb51 dataset has been taken for the study. This dataset provided has about 52 actions. But for the study, different action types are taken namely sword, ride bike, sword exercise, wave, cartwheel, and drink. These are split into target and non-target video directories. To obtain the features effectively, a pre-trained model namely r3d_18 is used to gather the output at specific layers. Then, for the videos depending on the STIP (interest points) HoG and HoF are constructed. For better visualization of the video frames a color histogram is featured to gather the different spatial orientations of different parts of the same size. As a result, for all the videos the feature descriptors are constructed for inputs. As a result, for any given video the nearest 10 videos are gathered.

Keywords: r3d_18, OpenCV, tensor, clustering, manhattan distance, euclidean distance, histogram, STIP, HoG, HoF.

1. Introduction:

In the digital era, the number of videos that are being captured has increased rapidly because of the increase in the use of mobiles, laptops, and tablets. With that rapid increase, the volume of videos has been increasing exponentially. So, it is important to understand the content of the video structure, specifically the action that was being performed in the video. The project focuses on advanced techniques for video analysis. We analyze different approaches to vector modeling, similarity measurement, and feature extraction in video data. The primary goals consist of:

1. Using the R3D-18 architecture to implement feature descriptors based on neural networks.
2. Using HoG and HoF descriptors, develop a Spatio-Temporal Interest Point (STIP) analysis.
3. Converting video frames into color histograms.
4. Taking a set of target videos' feature descriptors and storing them.
5. Putting in place a system for comparing similar videos to recognize and display them.

We use libraries like PyTorch, torchvision, Pyav, and OpenCV for deep learning and visual information extraction. With potential applications in duplicate detection and video recommendation systems, the project aims to improve the field of content-based video access and evaluation.

2. Description of the Proposed Solution:

2.1 Task - 0:

The entire dataset as well as the STIP files is downloaded from the source website[https://serre-lab.clps.brown.edu/resource/hmdb-a-large-human-motion-database/]. The .rar files are unrarred using tools like WinRar or online unrarring tools[https://www.ezyzip.com/open-extract-rar-file-online.html]. The extracted folders are further subdivided into folders named "Target_video" containing the specified 6 action folders and "Non_Target_video" containing the other 45 action videos.

2.2 Task - 1:

As a part of task -1, the task is to register a hook for layers namely layer3, layer4, and avgpool layers of the r3d_18 pre-trained network. For a given video file that is part of an action, the video frames are gathered for the specific video using OpenCV, and the videos that are being processed are captured to visualize for better human readability. Read the entire videos and append them to a list to get the total number of frames if frames are less than 32 the respective videos are skipped. Once the entire video is processed, we then need to select 32 video frames from the video frames and skip the videos that have frames less than 32. Selecting the first 32 frames causes various issues as there may be issues like the action not being captured which may result in gathering the video frames that are not required and eventually don't contain any features, Here, one of the issues that need to be addressed was the problem with selecting of frames the frames that are selected should possess the action of the video. But to obtain this, we need an approach that selects the frames from the entire video (OpenAI, 2024). To make this happen, a sliding window approach was chosen (You et al., 2019) where the window size was kept to 32. From each of the 32 frames, the frames are gathered. Once, the output is gathered from all the slides, the transformation is applied that resizes the frames and stacks them to get the required channels. Since the sliding window approach has many video frames depending on the total frames. From the processed frames max-pooling is applied to gather the maximum values from the frames and then take the mean from it to obtain a 512 size of tensors.

2.3 Task - 2:

Task 2 uses the STIP dataset to find insights into the given video dataset. We use the 72-dimensional HoG feature descriptor and 90-dimensional HoF feature descriptor to create cluster representatives from the top 400 confidence samples of each video file. We map an STIP sample of a video to the closest cluster out of the representatives and build a histogram for each of the 12 pairs of possible sigma and tau values. The distance from the cluster is calculated using Euclidean distance. This is concatenated to give us a bag of feature descriptors that are used to compare and find similar videos.

2.4 Task - 3:

Task 3 is to create histograms for the video inputs. Individual video frames are gathered using the OpenCV module so that we are able to extract the 3 required frames for the entire computation process. We extract the first, the middle, and the last frame from each video. For each frame, we create a 4x4 grid and each grid will have its own histogram bin. The first approach consists of considering 12 colors ('Yellow' ,'black', 'red', 'Purple', 'Green', 'Brown', 'Blue', 'Magenta', 'Silver', 'Lavender', 'Maroon', 'Grey'). Total of 16 histograms for each frame, 48 histograms for each video are extracted and saved in PNG format. This method helps in visual understanding of the histogram and its corresponding frame but takes huge computation time as well as not feasible to be used for further tasks. Approach 2 is to create csv files of each frame. There are 12 bins(12 different features or colors,similar to the previous approach) and 16 rows(Open AI, 2024[2]). histogram vectors are stored which helps in faster computation as well as usability for the further tasks.

2.4 Task - 4:

This task expands to getting all the features extracted from the input("Target_video" folder) using the methodology in each of the above tasks:

Task 1: As a part of this task the specific features descriptors that are gathered as a part of task -1 are appended to a list and then for each action in the target video, the outputs from each of the layers are gathered and they are stored as a json for faster extraction.

Task 3: Using the 2 approaches in task 3, we get results in different formats:

1. Approach 1 results in creating subfolders for each video. The folder contains the entire frame, each block of the 4x4 grid, as well as its corresponding histogram. All the above mentioned files are in the .PNG format. Hence further computation using them would not be feasible.
2. Approach 2 results in creating csv files for each frame histogram. We get the output in the format of "{video_name}_histogram_frame_{frame_id}.csv". Each csv has 12 columns and 16 rows(excluding the header) to store the histograms in vector format.

Task 2: We find the bag of feature descriptors for the target videos by finding the closest cluster representatives that were computed in Task 2 from the non-target videos. This gives us a single 480-dimensional vector to represent a video.

2.5 Task – 5:

Using task 1 and 4:

As a part of the previous task the outputs that are gathered as 'json' file are used for task -1. Since, as a part of task -1, the output of the target videos of different actions are stored. These are now imported and fed to a sample video to find "m" nearest member search. To implement this, the following algorithm has been adapted.

Algorithm:

1. To sample input video gather the respective outputs.
2. Append each of the layer outputs of the input video to an array which are represented as floating point numbers.

3. Calculate the Manhattan distance between the feature vectors of the input sample given to all those from the target video.
4. Append the distance measure to the result list, along with the video number for visualization along with a json file to visualize for that specific action.
5. Taking the distance measure as a sort parameter. Sort the entire list in ascending list.
6. From the list above, gather the nearest element that is ‘m’.
7. For the m nearest items, using OpenCV visualize the videos.

Manhattan Distance has been chosen as it computes the absolute distance between two features (Mior, 2020). That means, if both the features are similar the absolute distance may be equal to 0 or negligible. Manhattan Distance highlights the features and assists to recognize patterns (Mora-Garcia, Marti-Ciriquian, Perez-Sanchez, & Cespedes-Lopez, 2018). The sample output from the above tasks are the distance values and the visualization of the video that are similar to it using OpenCV.

Using task 3 and 4:

The code implements a video similarity comparison system using color histogram features. It loads histogram data from CSV files, computes similarity scores between a target video and other videos in a dataset, and identifies the most similar videos based on these scores.

Algorithm:

1. Load Histograms:
   a. Iterate through CSV files in the specified folder
   b. Extract video name and frame number from each file name
   c. Store histogram data in a nested dictionary: {video_name: {frame_number: histogram}}
2. Compute Similarity:
   a. For each video in the dataset (excluding the target video):
      - Compare corresponding frames (1 with 1, 2 with 2, 3 with 3)
      - Calculate similarity score using Euclidean metric.
      - Compute average score across all frames
   b. Store average similarity score for each video.
3. Find Most Similar Videos:
   a. Sort videos based on similarity scores.
   b. Select top 'm' videos with lowest scores (most similar).

Using task 2 and 4:

The bag of features is calculated for the 4 given samples using the cluster representatives from the non-target videos and this 480-dimensional feature is compared with those of the target videos using Euclidean distance to find the 10 most similar videos.

3. Interface Specification:

The goal of this project is to extract the features using a pre-trained neural network which actually produces 3 feature vectors. By using STIP features, 2 features are gathered namely HoG and HoF. Using color histogram, 1 feature is gathered. In total, there are 5 feature vectors that are gathered and the similarity between the given sample input and the output is calculated using different distance metrics. The interface of this project is like the user can provide the input video file. The given video file is processed and features are extracted. The nearest 'm' videos are selected for the give video. Using color histograms they can be visualized.

3.1 Obstacle Description:

Video data complexity: Compared to image-based feature extraction, video feature extraction is more complex due to the combination of spatial (static object) and temporal (movement) information. A successful solution must take into account both.

Efficiency and Accuracy: It can be computationally costly to extract features from a large number of frames. The system must strike a compromise between feature extraction accuracy and computational efficiency.

Feature Representation: A thorough representation of the video content may not be possible with basic feature extraction techniques. A combination of HoG, HoF, and deep learning-based features is required to achieve better accuracy.

Similarity Detection: Following feature extraction, an effective technique for distinguishing between and determining shared elements across videos is needed. Defining videos in a similar feature space that captures their time-based and spatial features is the difficult part.

4. System requirements/installation and execution Instructions:

4.1 System Requirements:

1. CPU: 8 Core CPU
2. RAM: 8GB
3. GPU: Not Essentially used.

4.2 SOL Requirements:

1. Nodes: 2 (Jupyter Notebook)

4.3 Programming Language:

1. Python

4.4 Modules Required:

1. OS
2. JSON

4.5 Libraries:

1. Pytorch
2. OpenCV
3. scikit-learn
4. Numpy
5. CSV
6. Matplotlib
7. Av
8. prettytable

5. Related Work:

In (Yoon & Han, 2022), the authors of this paper worked on 'Content Based Video Retrieval', to extract the features models the authors utilized 3D-CNN as their pre-trained neural network, the main choice of this approach as this would benefit the developers when working on limited resources. But even, through this approach when dimensionality reduction happens there would be a feature loss. So, the authors used network representation (Snell, Swersky, & Zemel, 2017) to eliminate feature loss.

6. Results:

The input videos that are provided using the feature vectors that are obtained are compared to those of all the target videos and the distance measured is printed. Since the features are vectors, the output from them is printed as a distance measure but using the video that is most similar to the sample input the video is visualized in Fig.1.

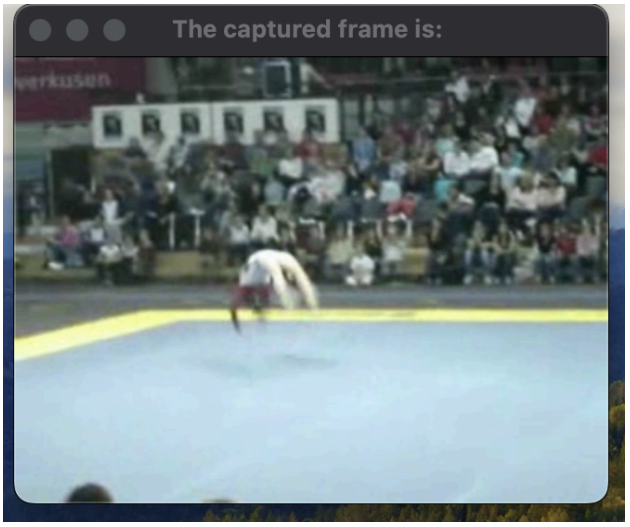

Fig-1: Visualization of Video

The above image corresponds to the video sample of the first sample input. In the corresponding m similar videos, the first near video consists of the action cartwheel which means the features that correspond to the cartwheel are extracted and are accurately being mapped.

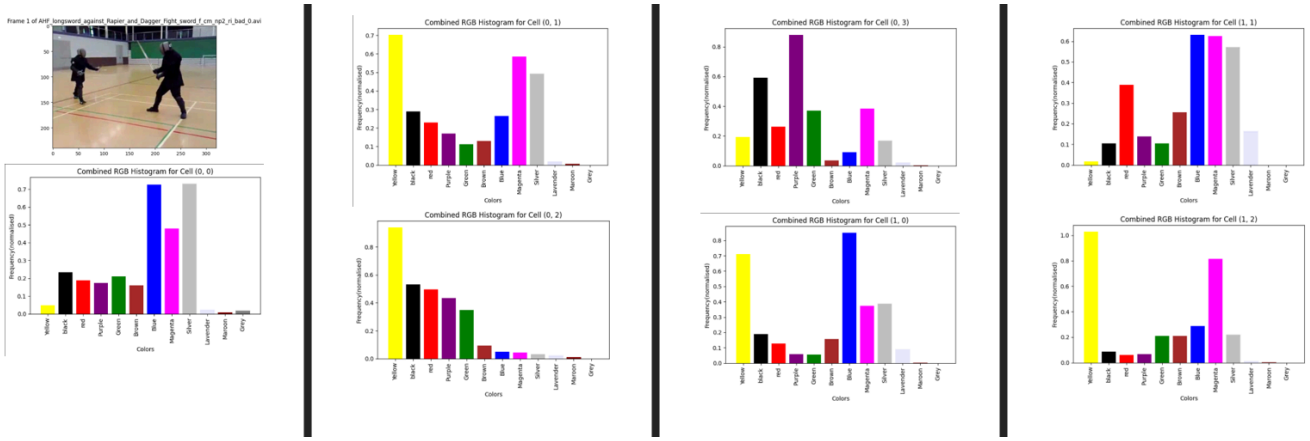

Fig-2: Histogram outputs for the first 7 out of 16 blocks

As we can see in the above snippet(Fig-2), we can infer from it how the histograms are being obtained. The image frame(top left) is divided into 16 blocks, and we are finding the histograms for each block. The color scheme chosen is based on a trial and error method so that most of the color spectrum of the image is covered while keeping minimum color redundancy.

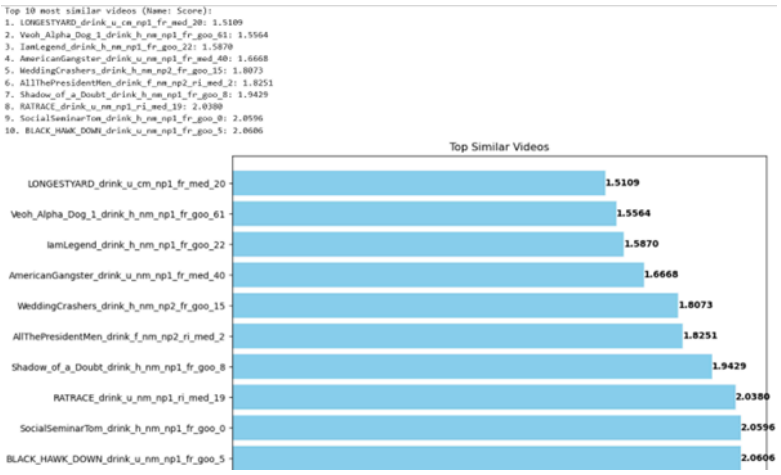

Fig-3: Similarity using histogram csv

Based on the similarity scores obtained by the comparison of target video histogram CSVs and those in the database, we are able to find the 10 most similar videos based on the Euclidean Distance. The above figure(Fig-4) is the output for the given input sample file.

HoG Closest Videos
STIP/target_videos/cartwheel/Bodenturnen_2004_cartwheel_f_cm_np1_le_med_0.avi.txt

Rank	Video Name	Distance
1	cartwheel\Bodenturnen_2004_cartwheel_f_cm_np1_ri_med_2.avi.txt	24.71841418861655
2	cartwheel\Bodenturnen_2004_cartwheel_f_cm_np1_le_med_5.avi.txt	29.13760456866693
3	cartwheel\Bodenturnen_2004_cartwheel_f_cm_np1_le_med_4.avi.txt	32.43454948045371
4	cartwheel\Sabine_en_Lieke_Turnen_3_cartwheel_f_cm_np2_fr_med_0.avi.txt	38.0
5	wave\CastAway1_wave_u_cm_np1_ri_med_4.avi.txt	38.62641583165593
6	ride_bike\Justin_lernt_Fahrrad_fahren_ride_bike_f_cm_np1_ri_med_2.avi.txt	44.59820624195516
7	cartwheel\cartwheel_challange_cartwheel_f_cm_np2_le_med_0.avi.txt	44.83302354291979
8	sword_exercise\HIGHLANDER_III_-_LOREENA_MCKENNITT_-_BONNY_PORTMORE_(HIGHLANDER_SCENES)_sword_exercise_u_cm_np1_fr_med_1.avi.txt	45.80392996239515
9	cartwheel\Lynn_en_Chelsea_Turnen_cartwheel_f_cm_np2_le_med_0.avi.txt	48.94895300208166
10	sword\Blade_Vs_Deacon_Frost_Sword_Fight_Scene_sword_f_cm_np2_le_med_2.avi.txt	49.09175083453431

Fig-4: Similarity based on BOF-HOG-480 descriptor

HoF Closest Videos
STIP/target_videos/cartwheel/Bodenturnen_2004_cartwheel_f_cm_np1_le_med_0.avi.txt

Rank	Video Name	Distance
1	cartwheel\Bodenturnen_2004_cartwheel_f_cm_np1_le_med_4.avi.txt	43.58898943540674
2	cartwheel\Bodenturnen_2004_cartwheel_f_cm_np1_le_med_5.avi.txt	47.78074926160116
3	ride_bike\lady_on_bike_ride_bike_f_cm_np1_ba_med_1.avi.txt	51.49757275833493
4	cartwheel\Geena_und_die_Radschl_ge___;-)_cartwheel_f_cm_np1_fr_med_2.avi.txt	51.526692111953004
5	ride_bike\1996_Tour_de_France_-_Indurain_Cracks_ride_bike_f_cm_np1_le_med_0.avi.txt	51.69139193328034
6	ride_bike\Tour_de_France_2009_ETAPA_16_ESPN_ESP__Kilometros_finales_ride_bike_f_cm_np1_fr_med_4.avi.txt	52.53570214625479
7	cartwheel\Geena_und_die_Radschl_ge___;-)_cartwheel_f_cm_np1_le_med_3.avi.txt	52.76362383309168
8	cartwheel\Jessica_and_Gregs_Cartwheel_Competition_cartwheel_f_cm_np1_le_med_4.avi.txt	52.78257288158659
9	ride_bike\Learn_How_To_Ride_A_Bike_at_Any_Age_ride_bike_f_cm_np1_ri_med_10.avi.txt	53.028294334251406
10	sword_exercise\Sword_and_Targe_Solo_Exercises_sword_exercise_u_cm_np1_le_med_8.avi.txt	55.154328932550705

Fig-5: Similarity based on BOF-HOF-480 descriptor

The above figures list the 10 most similar videos for the same given sample video based on their BOF-HOG-480 and BOF-HOF-480 descriptors. We can see that a majority of the files belong to the same action classification. However, the videos from the other classification also have similar distance metrics.

7. Conclusions:

From the above feature extraction techniques, the features are extracted for all the sample input videos and compared to those of all the target videos. using the similarity of the score, the nearest videos are gathered and visualized using OpenCV and histogram. So, by using feature extraction for the sample input it was understood that for a few of the actions the similarity was as expected however, there was a bit of overfitting that happened.

Bibliography:

1. You, D., Wu, X., Shen, L., Deng, S., Chen, Z., Ma, C., & Lian, Q. (2019). Online feature selection for streaming features using self-adaption sliding-window sampling. IEEE Access, 7, 16088-16100.
2. OpenAI. (2024). ChatGPT [Large language model]. https://chatgpt.com/c/66e7b073-d37c-800e-b0d9-4e4fd2ff4998
3. OpenAI. (2024)[2]. ChatGPT [Large language model]. https://chatgpt.com/share/66e8842f-0858-8003-a8d7-80289590659e
4. Mior, M. (2020, April 6). When would one use Manhattan distance as opposed to Euclidean distance? Data Science Stack Exchange. Retrieved September 13, 2024, from https://datascience.stackexchange.com/questions/20075/when-would-one-use-manhattan-distance-as-opposed-to-euclidean-distance
5. Mora-Garcia, R. T., Marti-Ciriquian, P., Perez-Sanchez, R., & Cespedes-Lopez, M. F. (2018). A comparative analysis of manhattan, euclidean and network distances. Why are network distances more useful to urban professionals?. International Multidisciplinary Scientific GeoConference: SGEM, 18(2.2), 3-10.
6. OpenCV documentation for Histograms. https://docs.opencv.org/4.x/d1/db7/tutorial_py_histogram_begins.html
7. Unrarring tool for Sol https://www.ezyzip.com/open-extract-rar-file-online.html
8. Dataset Source website https://serre-lab.clps.brown.edu/resource/hmdb-a-large-human-motion-database/
9. https://www.fusioncharts.com/blog/best-python-data-visualization-libraries/
10. https://stackoverflow.com/questions/4383571/importing-files-from-different-folder
11. https://www.geeksforgeeks.org/visualizing-colors-in-images-using-histogram-in-python/
12. https://stackoverflow.com/questions/33311153/python-extracting-and-saving-video-frames
13. Yoon, H., & Han, J. H. (2022). Content-based video retrieval with prototypes of deep features. IEEE Access, 10, 30730-30742.
14. Snell, J., Swersky, K., & Zemel, R. (2017). Prototypical networks for few-shot learning. Advances in neural information processing systems, 30.

Appendix:

1. Abhishek Mulasi: Discussion meetings, completed task 2, Used learning from task 2 to complete tasks 4 and 5, Perform the documentation and report completion
2. Sri Rupin Potula: Discussion meetings, completed task 1, Used learning from task 1 to complete tasks 4 and 5, Perform the documentation and report completion.
3. Vibha Swaminathan: Discussion meetings.
4. Vibhu Dixit: Discussion meetings, completed task 3, Used learning from task 3 to complete tasks 4 and 5, Perform the documentation and report completion.